Day 97

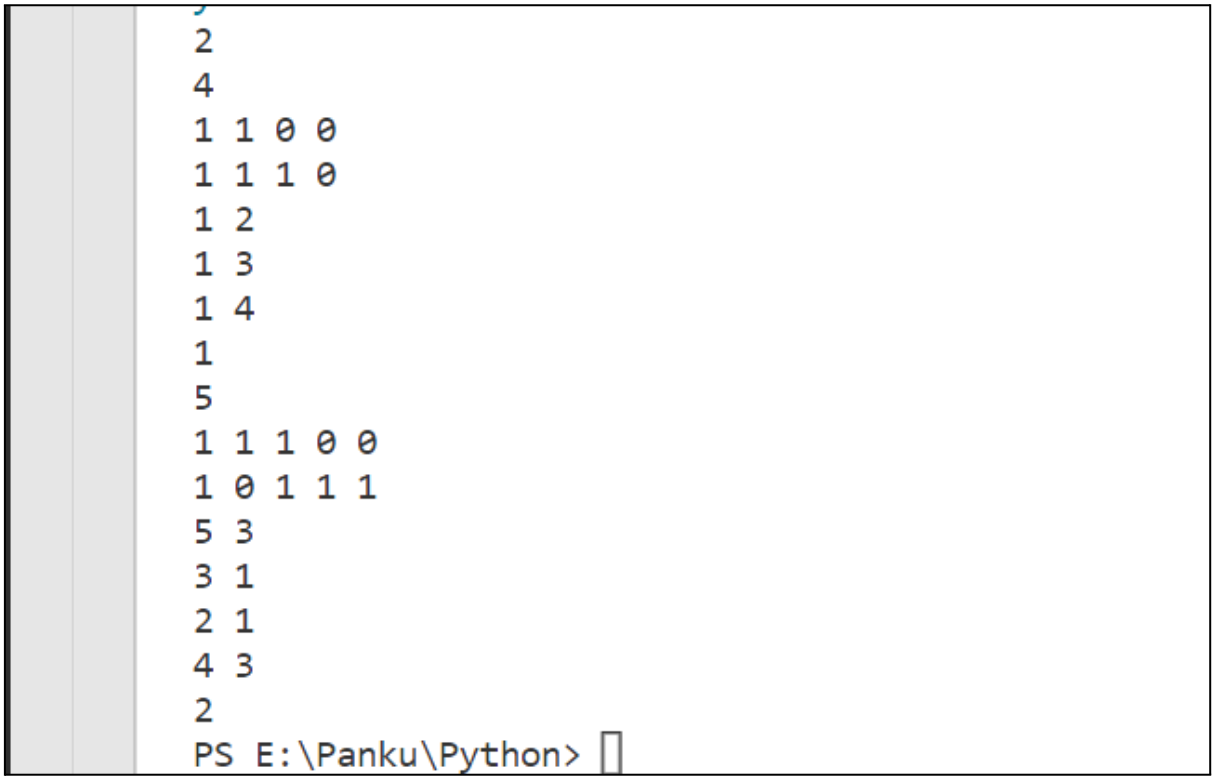
Q. Arun has a rooted tree of N vertices rooted at vertex 1. Each vertex can either be coloured black or white. Initially, the vertices are coloured A1?,A2?,…AN?, where Ai? ∈ {0,1} denotes the colour of the i-th vertex (here 0 represents white and 1 represents black). He wants to perform some operations to change the colouring of the vertices to B1?,B2?,…BN? respectively. Arun can perform the following operation any number of times. In one operation, he can choose any subtree and either paint all its vertices white or all its vertices black. Help Arun find the minimum number of operations required to change the colouring of the vertices to B1?,B2?,…BN? respectively.

main.py

```
from sys import stdin, setrecursionlimit

input = stdin.readline
setrecursionlimit(int(1e9))

def ii():
    return int(input())

def li():
    return list(map(int, input().split()))

a = []
b = []
n = 0
g = []
dp = []

def dfs(at, par, col):
    global a,b,g,dp
    if dp[col][at] != -1:
        return dp[col][at]

    if col == 2:
        if a[at] != b[at]:
            dp[col][at] = 1

            for ch in g[at]:
                if ch == par:
                    continue
                dp[col][at] += dfs(ch, at, b[at])

            return dp[col][at]

        dp[col][at] = 1
```

```
        for ch in g[at]:
            if ch == par:
                continue

            dp[col][at] += dfs(ch, at, b[at])

        res = 0

        for ch in g[at]:
            if ch == par:
                continue

            res += dfs(ch, at, 2)
        dp[col][at] = min(dp[col][at], res)

        return dp[col][at]

    dp[col][at] = (col!=b[at])

    for ch in g[at]:
        if ch == par:
            continue

        dp[col][at] += dfs(ch, at, b[at])
    return dp[col][at]

for _ in range(ii()):
    n = ii()
    a = [0] + li()
    b = [0] + li()

    g = [[] for i in range(n+1)]
    dp = []

    dp.append([-1 for i in range(n+1)])
    dp.append([-1 for i in range(n + 1)])
    dp.append([-1 for i in range(n + 1)])

    for i in range(n-1):
        x,y = li()
        g[x].append(y)
        g[y].append(x)

    print(dfs(1,0,2))
```

output


2
4
1 1 0 0
1 1 1 0
1 2
1 3
1 4
1
5
1 1 1 0 0
1 0 1 1 1
5 3
3 1
2 1
4 3
2
PS E:\Panku\Python>